IN THE SPECIFICATION

Please replace the paragraph beginning at page 3, line 14 with the following rewritten paragraph:

FIG. 1 illustrates a network computing environment in which aspects of the invention are implemented. A client computer 2 and server computer [4] 6 communicate over a network [6] 4, such as a Local Area Network (LAN), Wide Area Network (WAN), Storage Area Network (SAN), the Internet, an Intranet, etc., using a network protocol known in the art, e.g., Ethernet, Fibre Channel, TCP/IP, HyperText Transfer Protocol (HTTP), File Transfer Protocol (FTP), Fibre Channel, etc. The client includes a browser program 8, such as an HTML browser capable of downloading and rendering a page 10 of content from the server [4] 6 using a network transfer protocol, such as HTTP, etc. In certain described implementations, the page 10 includes multiple frames 12a...k, i.e., independent regions or subwindows of the page 10, where the content in each frame can be updated independently of other frames. For instance, the frames 12a...k may comprise HTML frames which display content from different web pages, i.e., content at different Universal Resource Locator (URL) addresses, in the same main browser 8 window. The client 2 may comprise any computing device known in the art, such as a personal computer, workstation, laptop computer, hand held computer, telephony device, mainframe, server, etc. The server 6 comprises a server-class machine or any other type of computing device capable of responding to data requests from the client 2.

Please replace the paragraph beginning at page 4, line 19 with the following rewritten paragraph:

A main servlet 20 handles the initial requests from the client 2 for the page 10. In response, the main servlet 20 generates a client session object 22a...n and an update queue 24a...n for each client session object 22a...n. Thus, one client session object 22a...n is maintained for each client 2 requesting the page of status and property information from the server 6. Each update queue 24a...n includes an array of queues, where each queue is used to store updates to the dynamic data within one of the frames

12a...k. A servlet 26a...m is provided for each type of component being monitored, such as a type of database or data source or a type of device, e.g., a specific type of storage device, switch, field replaceable unit (FRU), etc. Each servlet 26a...m generates one or more event listener objects 28a...j for each instance of the component type monitored by the servlet [28a...j] 26a...m. For instance, if a servlet [26a..m] 26a...m is associated with a particular type of storage device, e.g., a particular Redundant Array of Independent Disks (RAID) storage system, then one event listener object could be created for each instance of that particular type of RAID system to allow for independent monitoring of events at different instances of the component type. The event listener objects may be a member of an event listener Java class.

Please replace the paragraph beginning at page 5, line 8 with the following rewritten paragraph:

An event generator object 30a...j is instantiated to monitor states at one instance of a particular type of component 32a...j managed by the server [2] 6. The component instances 32a...j may comprise data sources, e.g., databases within the server 6 platform, or may comprise data sources or devices separate from the server 6. Upon detecting a state change, the event generator objects 30a...j send a notification to any registered event listener objects 28a...j including parameters describing the state change. The event generator objects may be a member of an event generator Java class.

Please replace the paragraph beginning at page 5, line 17 with the following rewritten paragraph:

In certain implementations, the components 32a...j monitored by the server [2] 6 may comprise field replaceable units (FRUs) and the events monitored may comprise the status and properties of different monitored FRUs. The page 10 displayed by the browser 8 may include a navigation pane which allows the user to select particular component instances to receive status and property information. Further, the user may select a property of a selected component to monitor, such as the selected component status. The

component or FRU status may indicate that the component is in the process of being discovered, discovered but not responding, that the component features are degraded and may require intervention, discovered successfully, component not discovered, component has failed, component has been discovered and functions properly, discovery pending, etc. For instance, if the monitored component is a disk system, then the monitored properties may include disk status, loop status, disk location, disk capacity, world wide name, product ID, etc. Further details of components and their status and properties that may be monitored are described in the "Sun StorEdge Component Manager 2.1 User's Guide", published by Sun Microsystems, Inc. (July 2000), which publication is incorporated herein by reference in its entirety.

Please replace the paragraph beginning at page 7, line 8 with the following rewritten paragraph:

FIG. 5 illustrates the event listener objects 28a...j as including registered sessions 80, which comprises a list of all client session objects [22a..n] 22a...n registered to receive updates to status and property information for the component instance associated with the event listener object 28a...j. If a component type is predefined as always associated with a particular frame 12a...k, then the event listener object 28a...j would place status and property updates in the queue 60a...n associated with that component type. Alternatively, in implementations where the user may customize the component instances represented in the frames [12a..b] 12a...k, then the frames 12a...k in different client systems may be associated with different component types. In such case, the registered sessions 80 in the event servlet listener 28a....j would indicate both the client session 24a...n and a queue 60a...n in the update queue array 22a...n in which the updates for that event listener 28a...j are placed.

Please replace the paragraph beginning at page 7, line 19 with the following rewritten paragraph:

FIGs. 6-11 illustrate logic implemented in the browser [2] 8 and server 6 architecture to provide status and property information to the browser 8. FIG. 6

illustrates logic implemented in the main servlet 20 to respond to an initial request for the page 10 at block 100. In response, the main servlet 20 creates (at block 102) a client session object 22a...n for the requesting browser 8 and an associated update queue array 24a...n including one queue 60a...k for each frame 12a...k in the page 10. As discussed, each frame 12a...k and corresponding queue 60a...k may be permanently associated with a particular type of component, e.g., FRU, disk, switch, etc., or the user may customize how frames 12a...k displayed in the browser page [8] 10 are associated with component types. The main servlet 20 returns (at block 104) the initial page [8] 10 to the browser 8 over the network 4, including a navigation pane 50 for each component type associated with the frame 12a...k. The navigation panes 50 may display all the component instances 32a...j monitored by the server 6.

Please replace the paragraph beginning at page 8, line 5 with the following rewritten paragraph:

At block 110, the browser 8 receives the page 10 including the navigation panes 50 for each frame 12a...k. In response, the browser 8 renders (at block 112) the entire page [8] 10 including a navigation pane 50 in each frame 12a...k through which the user may select one component instance 32a...j of the component type associated with the frame.

Please replace the paragraph beginning at page 8, line 21 with the following rewritten paragraph:

FIG. 8 illustrates logic at blocks 150 to 168 implemented in the servlet 26a...m in response to receiving the GET request for a user selected component at block 150. If (at block 152) an event listener object 28a...j is not already instantiated for the user selected component, then the servlet 26a...m instantiates (at block 154) one event listener object for the requested component. From block 152 or 154, the servlet 26a...m registers the client session object ID of the client 2 initiating the GET request in the registered sessions 80 (FIG. 5) of the event listener object 28a...j. At block 158, the servlet

[28a...m] 26a...m determines the target queue 60a...k (FIG. 3) in the update queue array 24a....n assigned to the browser 8 that will hold content updates for the requested component instance. As discussed, frames may be predefined as associated with a particular queue and component type, or the user may dynamically configure frames to be associated with a particular component type.

Please replace the paragraph beginning at page 10, line 11 with the following rewritten paragraph:

FIG. 9 illustrates logic implemented in the event listener objects 28a...j to provide content updates 70 to the update queue arrays [24a...j] 24a...n. At block 200, one event listener object 28a...j receives notification from an associated event generator object 30a...j including the updated status information for the component instance 32a...j monitored by the event generator object 30a...j. The event generator objects 30a...j monitor the component instances 32a...j for status changes and generate messages to registered event listener objects 28a...j providing information on the status change. The event listener object 28a...j then submits (at block 202) a request to the servlet 26a...m for the monitored component instance, which would be the servlet 26a...m that instantiated the event listener object 28a...j, to generate the content update 70 for the updated status information. As discussed the content update 70 includes the content identifier 72 of the field in the application pane 52 for which the update is provided and the actual update data 74. Upon receiving (at block 204) the content update 70 from the servlet 26a...m, the event listener object 28a...j then performs a loop at blocks 206 through 218 for each client session object ID in the registered sessions 80 list.

Please replace the paragraph beginning at page 10, line 25 with the following rewritten paragraph:

At block 208, the event listener object 28a....j determines the target queue 60a...k in the update queue array 24a...n for the registered client session ID associated with the event listener object 28a...j, i.e., the queue 60a....k having an event listener object ID field

62a...k identifying the event listener object 28a...j providing the content update. A content update 70 can be stateless or stateful. A stateless update is one where the latest update is the only one used, such as a temperature or other status of a component. A stateful content update is one that is accumulated with other content updates of the same type to present a cumulative list of the update information, such as a running log of system activity. If (at block 210) the update data 74 is stateless, then the event listener object [28a...m] 28a...j determines (at block 212) whether there is a content update 70 in the target queue 60a...k having the same content identifier (ID) 72 as the content update 70 to add. If so, then the entry with the same content ID 72 is replaced (at block 214) with the new content update 70 to add. Otherwise, if the content update 70 to add is stateful (from the no branch of block 210) or there is no pending content update 70 having the same content ID 72 (from the no branch of block 212), then the new content update 70 is added (at block 216) at the end of the target queue 60a...k. From block 214 or 216, control proceeds (at block 218) back to block 206 if there are further client session IDs in the registered sessions 80 list.

Please replace the paragraph beginning at page 12, line 8 with the following rewritten paragraph:

FIG. 11 illustrates logic embedded in the code of the concatenated scripts to cause the browser 8 to update specific fields in application panes [72] 52 of the frames 12a...k with the update data 74 in the content updates 70 packaged into the concatenated scripts. Control begins at block 300 with the browser 8 receiving the concatenated scripts of content updates 70. The browser 8 then performs a loop at block 302 through 314 for each script *i* in the concatenated group of scripts. For each script *i,* the browser 8 determines (at block 304) the target frame 12a...k associated with script *i*. Each script *i* may include a frame ID indicating the frame in the browser 8 to apply the content updates 70 in the script. Alternatively, the ordering of the concatenated scripts may be used to determine the frame to receive the content updates 70 in the script, where an empty script is used to indicate that there are no updates for the corresponding frame 12a...k.

Please replace the paragraph beginning at page 12, line 19 with the following rewritten paragraph:

The browser 8 then performs a loop at block 306 through 316 for each content update *j* in script *i* to update the frame 12a...k associated with the queue 60a...k including the content updates 70 packaged into script *i*. At block 308, the browser 8 determines the field in the target frame 12a...k associated with the content ID 72 of content update *j*. The content ID may identify a property or status field displayed in the application pane 70 of the target frame 12a...k. The browser 8 then updates (at block 310) the determined field in the target frame with the update data 74 in content update *j*. In this way, the concatenated scripts include code to cause the browser 8 to apply updates to only those fields in the application panes [72] 52 for which new property or status information is available.

Please replace the paragraph beginning at page 13, line 6 with the following rewritten paragraph:

FIG. 13 illustrates an example of a graphical user interface page 400 displaying frames in accordance with implementations of the invention. The page 400 includes multiple HTML frames, including a navigation frame 402, a menu frame 404, a tab frame 406, an application frame 408, an alarm counter frame 410, an alarm tab frame 412 and an alarm log frame 414. Each of these frames 400, 402, 404, 406, 408, 410, 412, and 414 may receive data and updates from a separate servlet, such as servlets 26a...m in FIG. 1, in the manner described above. The navigation frame 402 displays the component instances the user may select to cause the display of status and properties for the selected component in the application frame 408, including graphical representations [of the component] 420 of the component and a property table 422. Selection of one of the hypertext links in the menu frame 404, tab frame 406, and alarm tab frame 412 would cause the display of another window including data displayed in one or more frames that have one or more underlying servlets in the server 6 providing data and updates in the manner described above.

Please replace the paragraph beginning at page 14, line 1 with the following rewritten paragraph:

With the above described architecture and logic, the server 6 transfers only the data needed to update those fields or areas of the page [8] 10 having new data. This technique optimizes network bandwidth and server load because only new updated data is transferred in response to a refresh request, instead of transferring the entire page content. Further, by minimizing the data transferred as part of a refresh update, the transfer time is optimized because the amount of data transferred is minimized, thereby reducing any delays in providing data updates to the page 8 in the browser. Yet further, the rendering operations by the browser 8 to update the page 10 frames 12a...k with the update data is minimized because the browser 8 need only update those fields in the page for which update data is supplied. The browser 8 does not have to redraw the entire page contents in the page.

Please replace the paragraph beginning at page 15, line 17 with the following rewritten paragraph:

The described implementations utilized only a single refresh request to obtain the updates for all the frames in the page. In alternative implementations, multiple refresh metatags may be embedded in the page to perform separate refresh operations [fo] for different frames, thereby not using a single refresh for all the frames.

Please replace the paragraph beginning at page 16, line 7 with the following rewritten paragraph:

In the described implementations, all of the objects and servlets used to provide updates to the client browser were described as implemented on a single [sever] server 6. In alternative implementations, the objects and servlets described herein may be implemented on different computer systems and servers according to a distributed application architecture known in the art. For instance, the servlets and objects may be implemented in the Sun Microsystem's JIRO architecture where distributed servlets and

objects are registered with a lookup service and communicate using proxy objects accessed through the lookup service.

Please replace the paragraph beginning at page 16, line 14 with the following rewritten paragraph:

In the described implementations, update data was provided to the client browser in response to GET requests submitted to the server 6, whereby the client pulls the updates from the server 6. In alternative implementations, the queued updates may be pushed from the server 6 to the client [6] 2 to render in the browser 8.